Amendments to the Claims

This listing of claims will replace all prior versions and listings of claims in the application:

Claim 1 (currently amended). A method for gathering data from memory of a computer system, comprising:

suspending execution of all threads executing on the computer system, including at least one thread in at least one real-time control program and not including ~~except for~~ a thread performing the method;

after suspending execution of the threads, following a plurality of memory element descriptors of a machine readable record list to locate data in the memory of the computer system, where each memory descriptor is descriptive of data to be retrieved from memory of the computer system;

gathering data specified by the plurality of memory element descriptors while maintaining data coherency and without disrupting operation of the at least one real-time control program;

restarting execution of the suspended threads after gathering the data; and

formatting the data into a buffer.

Claim 2 (original). The method for gathering data from memory of a computer system of Claim 1, wherein at least one memory descriptor is descriptive of a list memory type, including location information of a head of a list and tag information for at least one data element to be gathered from a node of the list.

Claim 3 (original). The method for gathering data from memory of a computer system of Claim 1, wherein at least one memory descriptor is descriptive of a scalar memory type.

Claim 4 (original). The method for gathering data from memory of a computer system of Claim 3, wherein at least one memory descriptor is descriptive of a list memory type, including location information of a head of a list and tag information for at least one data element to be gathered from a node of the list.

Claim 5 (original). The method for gathering data from memory of a computer system of Claim 3, wherein at least one memory descriptor is a list memory descriptor, including location information of a head of a first list, location information of a head of a second list in nodes of the first list, and tag information for at least one data element to be gathered from nodes of the second list.

Claim 6 (currently amended). A method for parsing a linked list to extract data therefrom, the linked list stored in memory of a computer system, comprising:

constructing a record list, the record list comprising at least a first list element descriptor descriptive of data to be retrieved from a first linked list;

following a list head locator of the list element descriptor to a head of the first linked list;

stopping execution of all threads executing on the computer system, including at least one thread in at least one real-time control program and not including ~~except for~~ a thread parsing the list;

following links of the head of the first linked list to a first node of the linked list;

interpreting at least one tag of the first list element descriptor to locate data of the node; and

extracting data from the node while maintaining data coherency and without disrupting operation of the at least one real-time control program; and

resuming execution of all threads stopped during the step of stopping execution.

Claim 7 (previously presented). The method of parsing a linked list of Claim 6, wherein:

the record list further comprises a second list element descriptor descriptive of data to be retrieved from a second linked list, and wherein a node of the first linked list contains a head of the second linked list; and

the method further comprises:

following a list head locator of the second list element descriptor to a second list head of the node of the first linked list;

following links of the second list head to a node of the second list;

interpreting at least one tag of the second list element descriptor to locate data of the node of the second list; and

extracting data from the node of the second list.

Claim 8 (original). The method of parsing a linked list of Claim 7, further comprising the step of formatting the extracted data into a capture buffer.

Claim 9 (canceled).

Claim 10 (previously presented). The method of Claim 9, further comprising:

following links of the node of the second list to a second node of the second list, and

extracting data from the second node of the second list

according to at least one tag of the second list element descriptor.

Claim 11 (currently amended). A computer system comprising at least one processor, a memory system, and computer readable code recorded within the memory system, the code comprising computer readable code for:

receiving a record list, the record list comprising at least a first list element descriptor descriptive of data to be retrieved from a first linked list;

following a list head locator of the list element descriptor to a head of the first linked list;

stopping execution of all threads executing on the computer system, including at least one thread in at least one real-time control program and not including a thread parsing the list;

following links of the head of the first linked list to a first node of the linked list;

interpreting at least one tag of the first list element descriptor to locate data of the node; ~~and~~

extracting data from the node while maintaining data coherency and without disrupting operation of the at least one real-time control program; and

resuming execution of all threads stopped during the step of stopping execution.

Claim 12 (previously presented). The computer system of Claim 11, wherein:

the record list further comprises a second list element descriptor descriptive of data to be retrieved from a second linked list, and wherein a node of the first linked list contains a head of the second linked list; and

the computer readable code further comprises computer readable code for:

following a list head locator of the second list element descriptor to a second list head of the node of the first linked list;

following links of the second list head to a node of the second list;

interpreting at least one tag of the second list element descriptor to locate data of the node of the second list; and

extracting data from the node of the second list.

Claim 13 (currently amended). A symbolic debugger for accessing data of named executable modules of an operating system executing on a target machine, the operating system having version information, the symbolic debugger comprising computer readable code stored on computer readable media, the computer readable code comprising code for:

a collection driver for execution on the target machine;

a user interface capable of coupling to the collection driver; and

a symbol resolution system capable of coupling to the user interface;

wherein the user interface comprises computer readable code for constructing an input record list containing records describing data to be captured, at least some records of the input record list containing information derived from symbols resolved by the symbol resolution system, and transmitting the input record list to the collection driver;

wherein the collection driver further comprises code for stopping execution of all threads executing on the computer system, including at least one thread in at least one real-time control program and not including ~~except~~ the collection driver;

wherein the collection driver further comprises code for interpreting the input record list and collecting operating system data into a capture buffer specified by the input record list while maintaining data coherency and without disrupting operation of the at least one real-time control program, and transmitting the capture buffer to the user interface; and

wherein the collection driver further comprises code for resuming execution of all previously-stopped threads.

Claim 14 (original). The symbolic debugger of Claim 13, wherein the collection driver is capable of interpreting a record of the input record list that specifies information to be gathered from multiple nodes of a linked list.

Claim 15 (original). The symbolic debugger of Claim 14, wherein the collection driver is capable of interpreting a record of the input record list that specifies information to be gathered from multiple nodes of a linked list having a list head located in a node of a parent list, a list head of the parent list being specified by a record of the input record list.

Claim 16 (original). The symbolic debugger of Claim 14, wherein the collection driver is capable of interpreting a record of the input record list that specifies scalar information to be gathered from designated locations of the memory system.

Claim 17 (original). The symbolic debugger of Claim 14, wherein the collection driver further comprises a communications interface capable of receiving the record list over a network connection and comprises computer readable code for reading the version information from the operating system executing on the target machine.